



Sécurité

ANF Qualité Logicielle

Sommaire



Who am I ?

Quand on parle de sécurité ...

Comment fabriquer un bon sandwich ?

Les bonnes pratiques

SAST / DAST / SCA / ...

C'était (presque) pas de votre faute !

Disclaimer



Cette présentation contient des parties interactives !

Merci d'attendre après le repas pour faire votre sieste





Who am I ?

Skipping before ...



2015



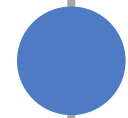
DevOps @ CCIN2P3



2019



Pentester



2022



COO @ DSecBypass mais surtout Pentester / Dev

Now !



Sécu-Riz-Thé ??????



- ▶ Ça vous parle ?
- ▶ Développement sécurisé ?
- ▶ Quelles sont vos contraintes lorsque vous développez ?
- ▶ Ça évoque quoi chez vous ?



Sécu-Riz-Thé ??????



Quelques constats :

“Je suis le seul à utiliser mon application / programme”

“10 ans qu’il fonctionne et aucun problème !”

“Mes utilisateurs sont gentils” 😊

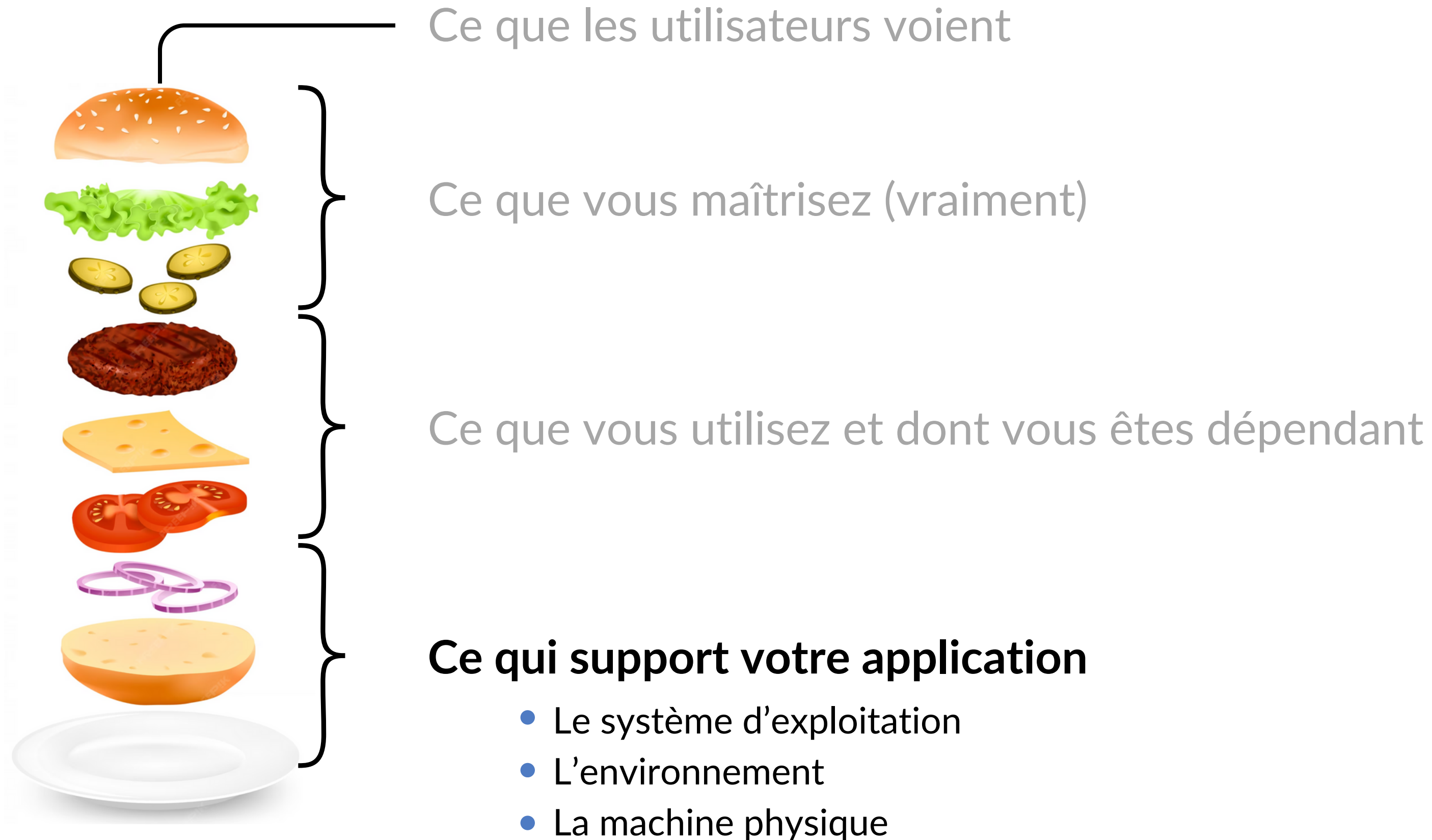
“Mon timing est très serré, je vais à l’essentiel”

“Je suis contraint par le langage à utiliser”

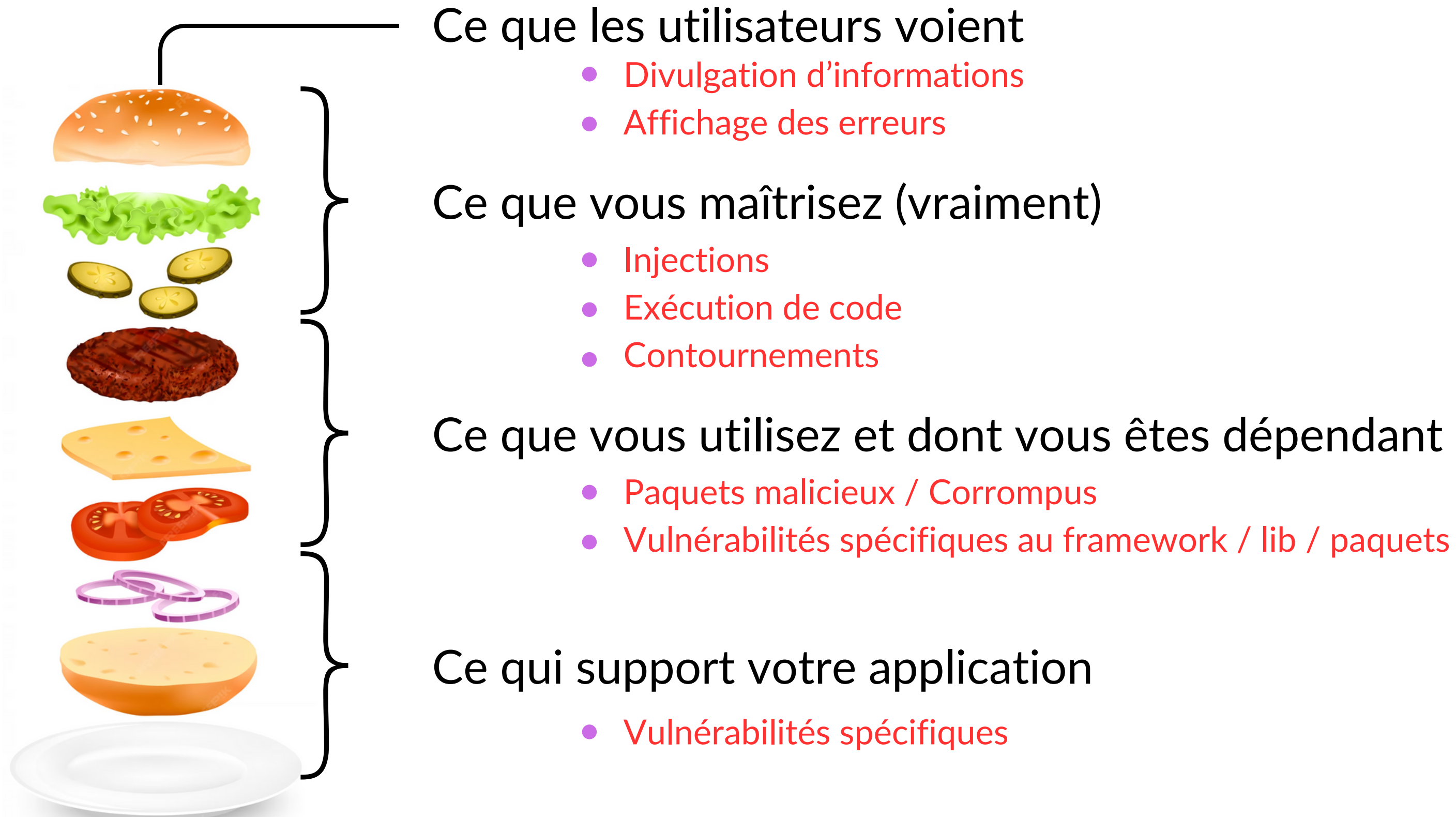
“Je suis expert sur cette techno et je fais de la veille !”



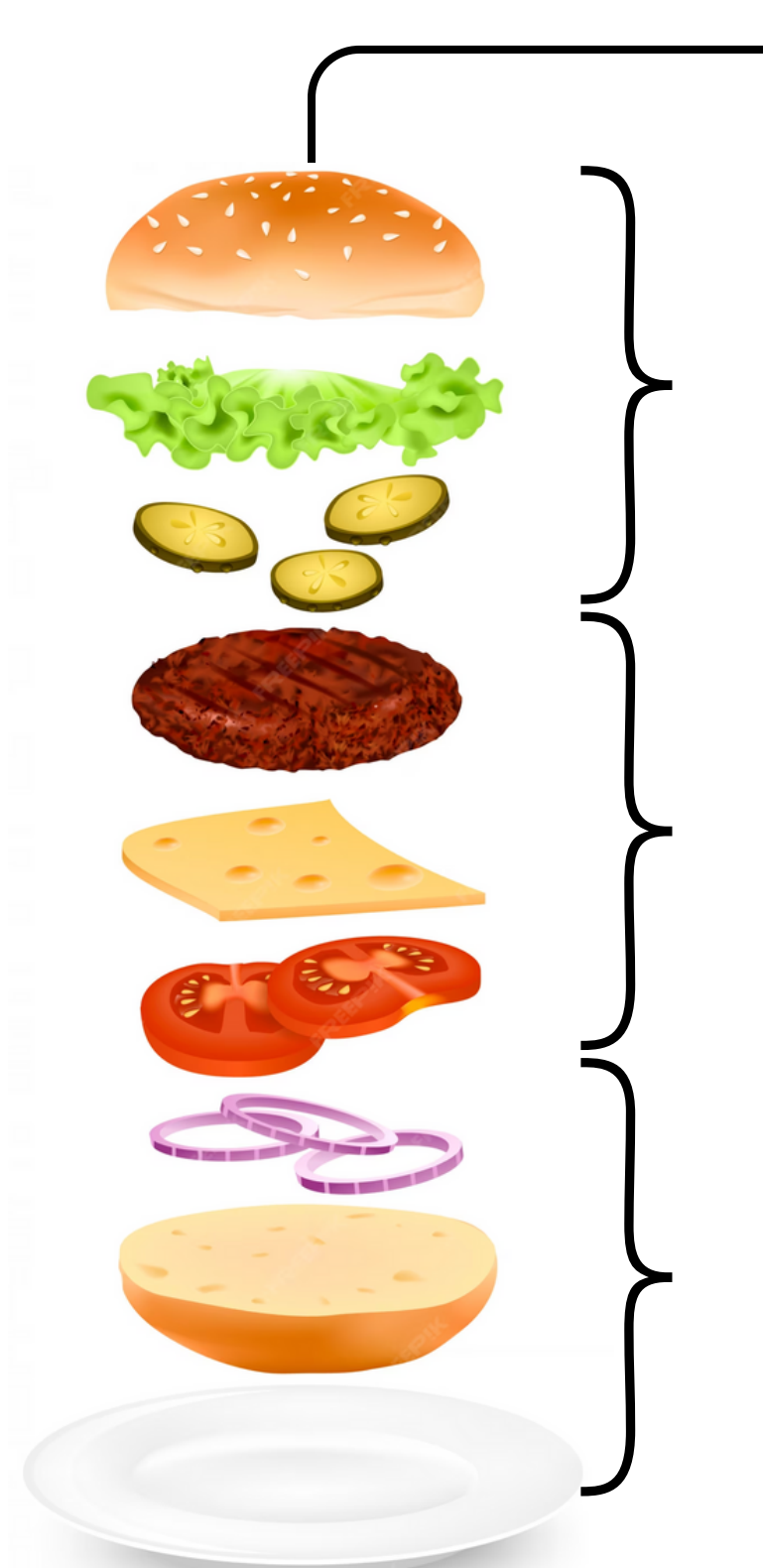
Recette d'un bon sandwich



mauvais Recette d'un ~~bon~~ sandwich



Recette d'un ^{super} bon sandwich



Ce que les utilisateurs voient

- ~~Divulgarion d'informations~~
- ~~Affichage des erreurs~~

Faites tester votre application

Ce que vous maîtrisez (vraiment)

- ~~Injections~~
- ~~Exécution de code~~
- ~~Contournements~~

Tenez-vous au courant des dernières implémentations en terme de sécurité

Ce que vous utilisez et dont vous êtes dépendant

- ~~Paquets malicieux / Corrompus~~
- ~~Vulnérabilités spécifiques au framework / lib / paquets~~

Faites de la veille, Suivez les mises à jour

Ce qui support votre application

- ~~Vulnérabilités spécifiques~~

Délégez cette partie aux équipes dédiées

Bonnes pratiques



Partez du principe que vous ne pourrez pas tout maîtriser



Suivez les produits / technologies que vous utilisez



Implementez les guides sécurité (quand ils existent ...)



Ne faites JAMAIS confiance aux utilisateurs



Testez, auditez votre code



Doutez (un peu) de vous

Gros mots



SAST

SCA

DAST

IAST



SAST

Static Application Security Testing

Qu'est-ce que c'est ?

- ? Méthodologie qui mélange outils et technologies
- ? “Test en boîte blanche”
- ? Analyse dite “au repos”

A quoi ça sert ?

- Faire respecter les formats et normes de programmations ★
- Vérifier la présence de vulnérabilités dans le code ★

SAST

Static Application Security Testing



Comment ça marche ?

- ⚙️ Recherche ligne par ligne, instruction par instruction et comparaison à une bibliothèque d'erreurs et de règles

A quoi il faut faire attention ?

Un langage = un SAST ⚠️

Ce qui est couvert en terme de tests ⚠️

Les faux-positifs ⚠️

SAST



Et pleins d'autres : [https://owasp.org/www-community/Source Code Analysis Tools](https://owasp.org/www-community/Source_Code_Analysis_Tools)









SCA

Software Composition Analysis

Ce qu'il faut retenir

-  Analyse tout le code que vous n'avez pas écrit (framework, lib, plugin, ...)
-  En partie fait par des outils SAST mais jamais aussi bien
-  Pas une analyse de code mais un rapport basé sur des listes pré-établies (CVE db, exploit DB, vendors, chercheurs indépendants, ...)
-  Très rapide !

Votre meilleur ennemi ? Les dépendances

SCA





DAST

Dynamic Application Security Testing

Qu'est-ce que c'est ?

- ? Ensemble d'outils et de techniques
- ? "Test en boîte noire"
- ? En cours d'exécution sur un environnement de test

A quoi ça sert ?

- Détecter les problèmes de configuration ★
- Identifier des vulnérabilités logiques ★

DAST



Dynamic Application Security Testing

Comment ça marche ?

- ⚙ Simule des entrées (inputs) selon des tests pré-conçus

A quoi il faut faire attention ?

Nécessite de très bien connaître l'application 

Time consuming 



IAST

Interactive Application Security Testing

Qu'est-ce que c'est ?

- ? Une combinaison du meilleur des tests SAST et DAST
- ? Méthode d'analyse d'une application reposant sur des interactions

A quoi ça sert ?

- Faire le lien entre la vulnérabilité logique et le code ★
- Apporter des corrections rapides et efficaces ★

IAST



Interactive Application Security Testing

Comment ça marche ?

- ⚙ Analyse du fonctionnement
- ⚙ Recherche de vulnérabilités
- ⚙ Contrôle de l'exécution

A quoi il faut faire attention ?

- Nécessite des connaissances en sécurité ⚠
- Chute des performances sur les segments IAST ⚠
- Ne teste pas la totalité de l'application ⚠

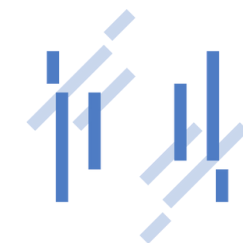
DAST / IAST



Et pleins d'autres : https://owasp.org/www-community/Vulnerability_Scanning_Tools



Workflow



Alors, on les place où ?

SCA

IAST

DAST

SAST

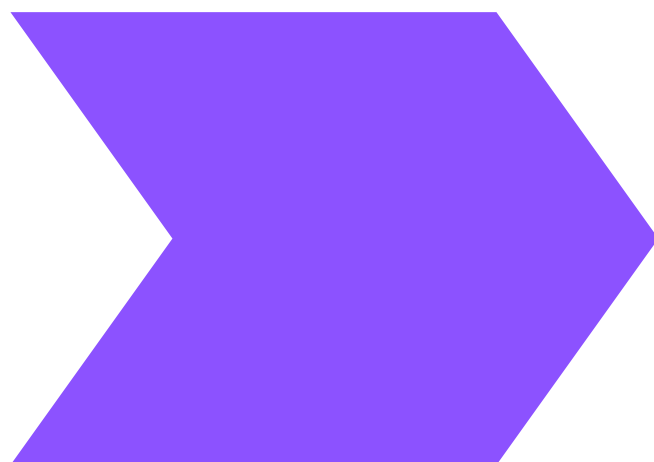
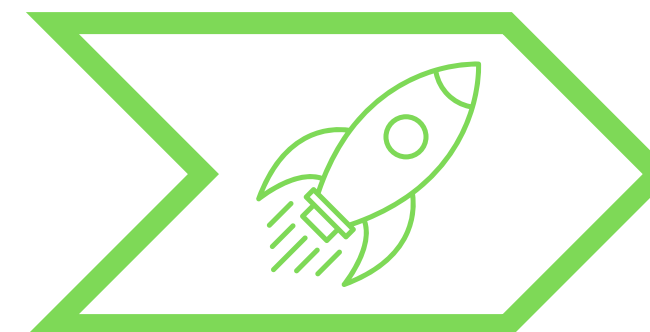
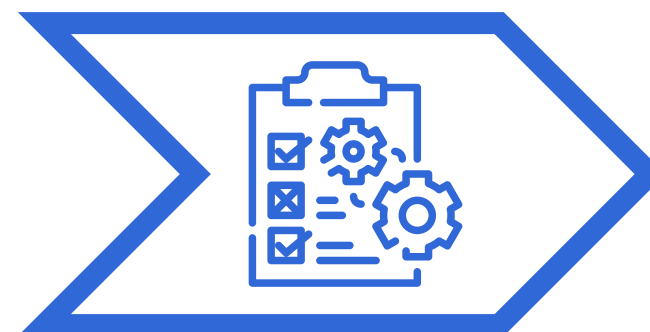
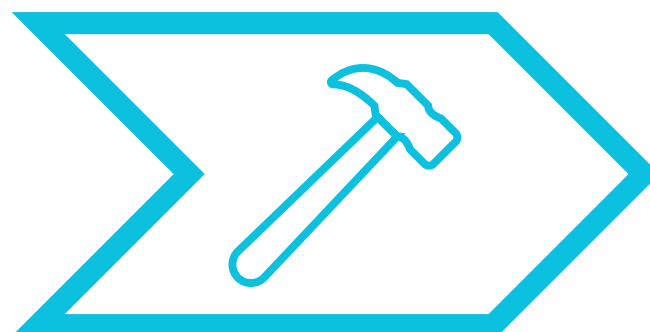
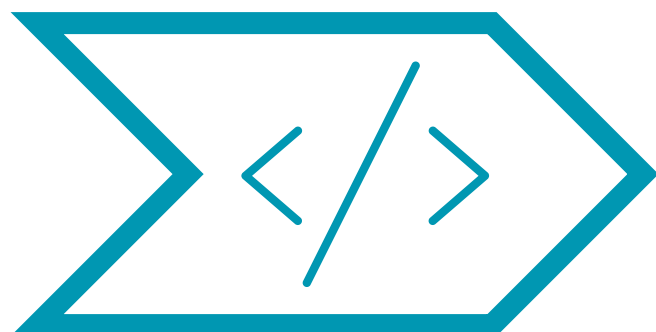
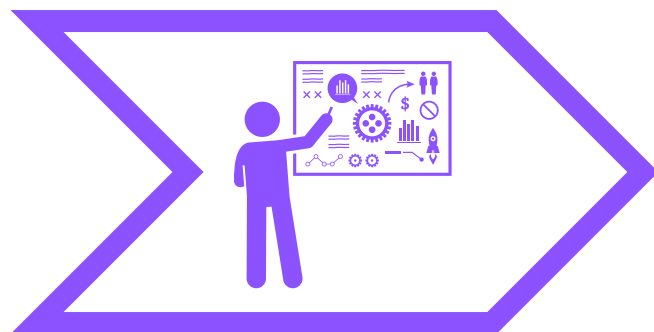
PLAN

CODE

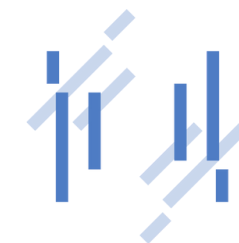
BUILD

TEST

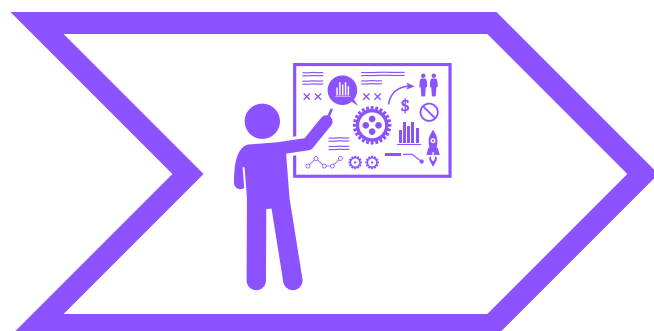
DEPLOY



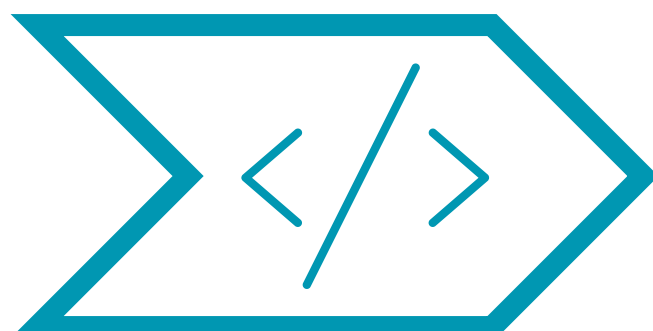
Workflow



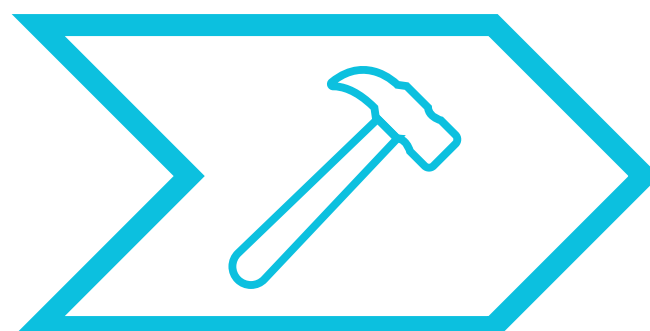
PLAN



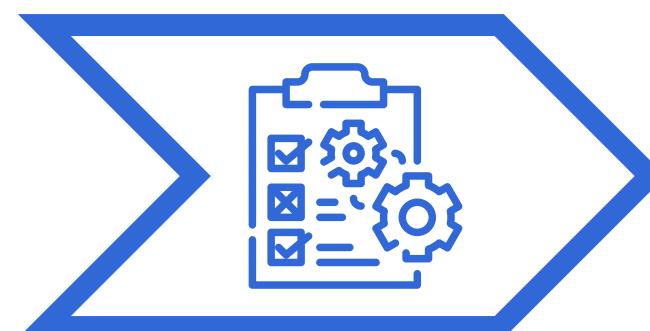
CODE



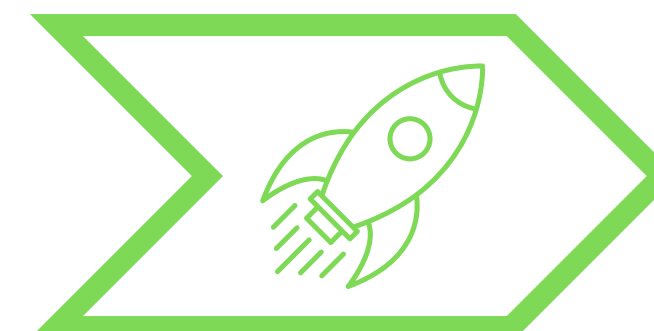
BUILD



TEST



DEPLOY



Dans la vraie vie



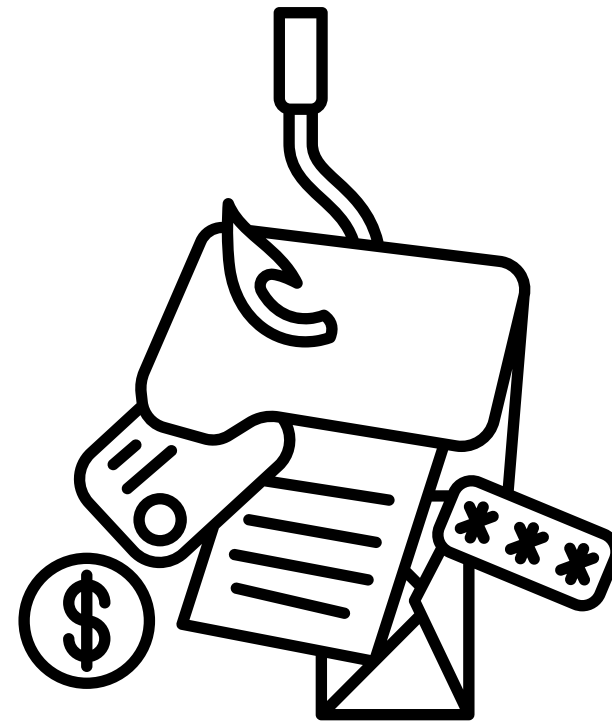
Vous avez une idée des problèmes qu'on peut rencontrer ?



C'était (presque) pas de votre faute !



Le phishing, ça vous parle ?



C'était (presque) pas de votre faute !

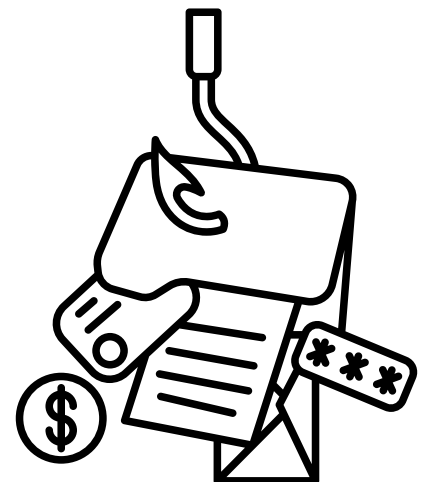
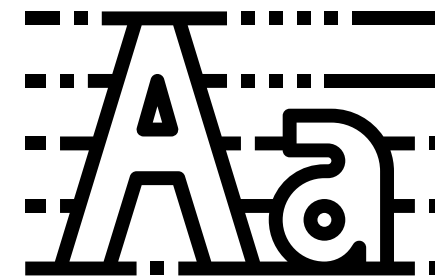


Le phishing, ça vous parle ?

Parmi les ingrédients, on retrouve le “faire croire que”, ainsi que pleins d'autres (urgence, code visuel, ...)

Imaginez que ça s'applique un peu partout

TYPO-SQUATTING



C'était (presque) pas de votre faute !

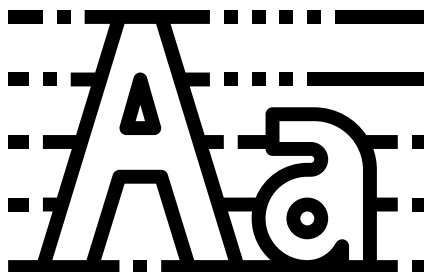


TYPO-SQUATTING

Faire croire que ce que vous voyez est ce à quoi vous pensez

Ce n'est pas de la magie ! ✨ ✨

C'est basé sur de la confusion et l'inattention 🧑🏻 🧑🏻



C'était (presque) pas de votre faute !



TYPO-SQUATTING

Faire croire que ce que vous voyez, est ce à quoi vous pensez

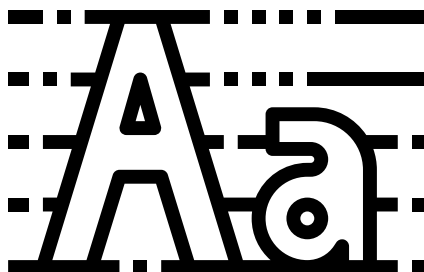
guoogle.com !

g00gle.com !

gogle.com !

✓ google.com

goog1e.com !



C'était (presque) pas de votre faute !



TYPO-SQUATTING

En 2019, deux paquets malicieux ont été identifiés sur PyPI.

Leur but était de dérober des clés SSH et GPG des machines infectées.

Leurs noms ?

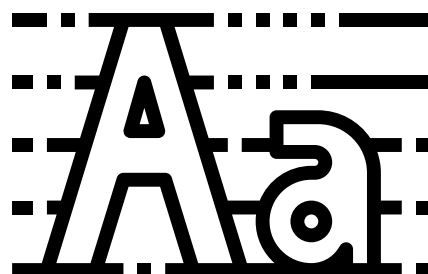
jellyfish pour jellyfish



JEILYFISH

<https://incolumitas.com/2016/06/08/typosquatting-package-managers/>

<https://snyk.io/fr/blog/malicious-packages-found-to-be-typo-squatting-in-pypi/>



C'était (presque) pas de votre faute !



TYPO-SQUATTING

En 2019, deux paquets malicieux ont été identifiés sur PyPI.

Leur but était de dérober des clés SSH et GPG des machines infectées.

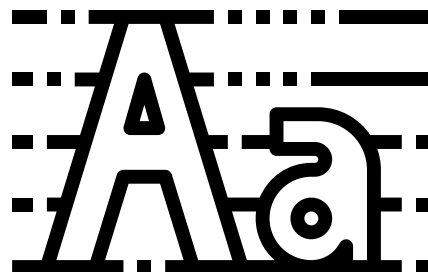
Leurs noms ?

jellyfish pour **jellyfish**

python3-dateutil pour **dateutil**

<https://incolumitas.com/2016/06/08/typosquatting-package-managers/>

<https://snyk.io/fr/blog/malicious-packages-found-to-be-typo-squatting-in-pypi/>



C'était (presque) pas de votre faute !



2021

Contexte : été 2020, Justin Gardner partage un morceau de code source Node.js trouvé sur GitHub et appartenant à Paypal

```
1  "dependencies": {
2    "express": "^4.3.0",
3    "dustjs-helpers": "~1.6.3",
4    "continuation-local-storage": "^3.1.0",
5    "pplogger": "^0.2",
6    "auth-paypal": "^2.0.0",
7    "wurfl-paypal": "^1.0.0",
8    "analytics-paypal": "~1.0.0"
9  }
```

<https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610>

C'était (presque) pas de votre faute !



2021

Vous avez vu le problème ? Alex Birsan, lui oui !

```
1  "dependencies": {
2    "express": "^4.3.0",
3    "dustjs-helpers": "~1.6.3",
4    "continuation-local-storage": "^3.1.0",
5    "pplogger": "^0.2",
6    "auth-paypal": "^2.0.0",
7    "wurfl-paypal": "^1.0.0",
8    "analytics-paypal": "~1.0.0"
9  }
```

<https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610>

C'était (presque) pas de votre faute !



2021

Vous avez vu le problème ? Alex Birsan, lui oui !

Etapes de la compromission:

- Création d'un paquet malicieux
- Exécution de code à chaque install via un script "preinstall"
- Exfiltration de données via DNS
- Réitérer !!!

<https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610>

Open- n'importe quoi



Votre fichier de config est dans votre repository de code ?



Les faits : Des clés d'API, des secrets sont publiés tous les jours sur des repos publics

Le problème : La recherche et l'identification est tellement simple que ça en devient ridicule

<https://news.sophos.com/en-us/2019/03/25/thousands-of-coders-are-leaving-their-crown-jewels-exposed-on-github/>

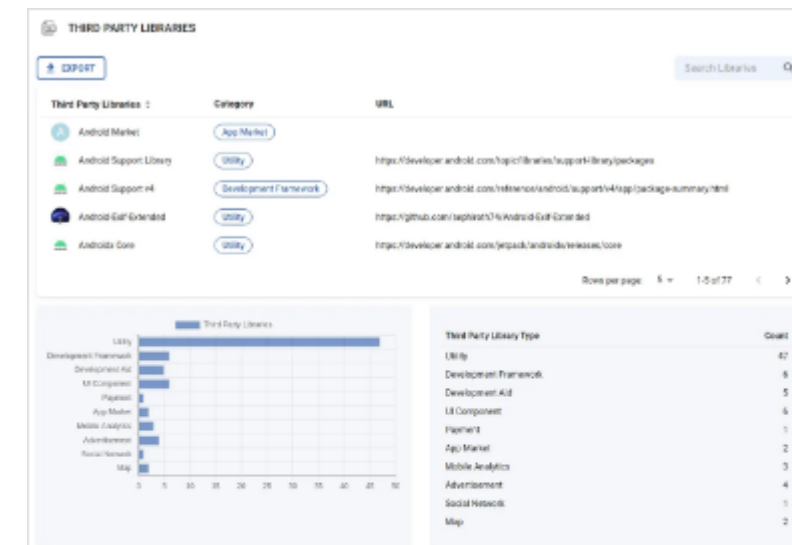
Open- n'importe quoi



Thousands of API and cryptographic keys leaking on GitHub every day

Researchers have found that one of the most popular source code repositories in the world is still housing thousands of publicly accessible user credentials.

2019



Exposed Payment Integration API Keys Imperil Millions of Users' Transaction Detail...

CloudSEK has observed that a wide range of companies — both large and small — that cater to millions of users have mobile apps with API keys that are hardcoded in...

2021



Starbucks Exposed An API Key In GitHub Public Repository

Recently, a researcher discovered a Starbucks API key exposed in a public...

Latest Hacking News (Jan 4, 2020)

2020

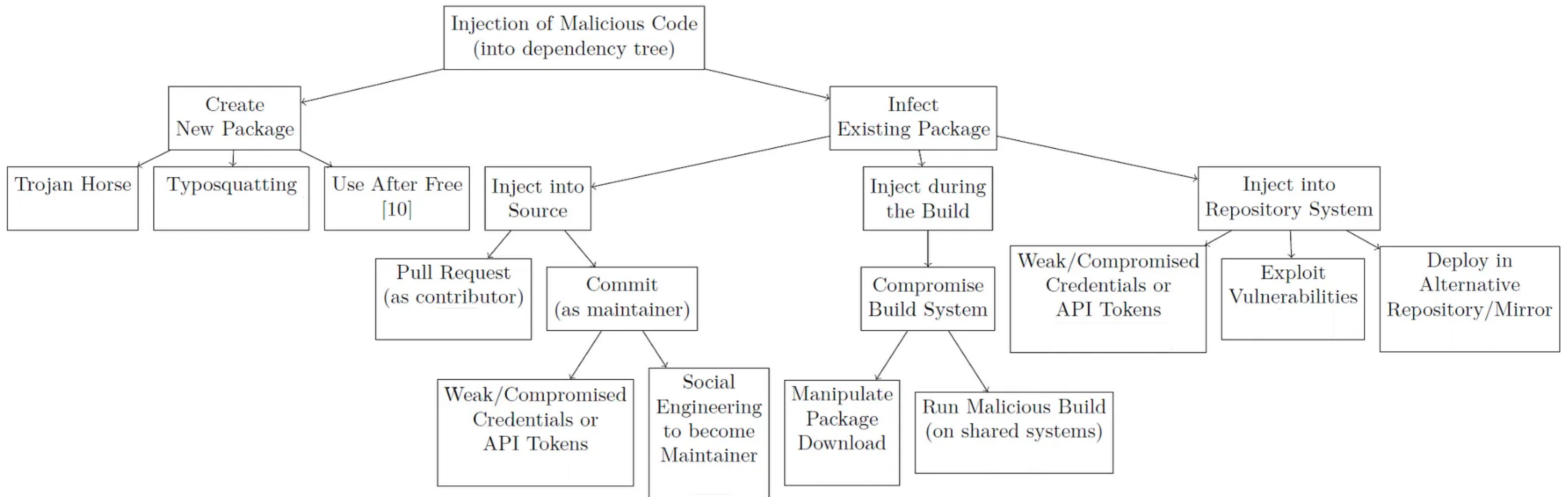
2022



Leaked Algolia API Keys Exposed Data of Millions of Users

Threat detection firm CloudSEK has identified thousands of applications leaking Algolia API keys, and tens of applications with hardcoded admin secrets, which could allow attackers to...

Le paysage



Questions ?



Merci !

 @nomekrax

 justin-bussery-pentester

 jbu@dsecbypass.com